B211 Internet Computing

Internals of the Web Server

Learning Objectives

1. Understand the basic mechanism used by HTTP to communication.

2. Understand the basic operations done by a web server.

Lecture Outline

- HTTP communications
  - URL
  - Format of request/response messages
- How does a web server work?
- Common Gateway Interface (CGI)
- Current web server usage statistics

Identifying Resources on the Web (the URL)

- Resources on the web, most commonly web pages, are identified using Uniform Resource Locators (URLs)
- URLs are similar to filenames, except they include:
  - the name of the server they are on
  - network protocols used to access them

Components of a URL

- A URL has the following general form:

  <scheme> : <scheme-specific name>

- Most schemes have the following formats:

  <protocol>://<user>:<password>@<host>:<port>/<path>

- When using common browsers, we usually only specify the host and path.
  - The protocols is assumed to be “http”, and the port number is “80”
  - The browser tacks on these assumed information to the requests.

Example URLs

http://www.w3.org/hypertext/WWW/TheProject.html

http://www.it.murdoch.edu.au:8888/~jsmith/index.html

ftp://hiew@ftp.it.murdoch.edu.au/pub/paper.txt

mailto:h.hiew@murdoch.edu.au

Some Common WWW Schemes

- http
  - access any web resources (html pages, text files, programs, ect)
- ftp
  - get or put a permanent copy of the required resource
- mailto
  - send an email message to the specified URL.
- file
  - access any file on the local file system (on the same machine/operating system as the web browser)

Hostnames in URLs

- By convention, domain names for hosts also usually includes the name of the protocol handled by server.

- Eg.
  - Murdoch IT’s ftp server is called ftp.it.murdoch.edu.au, and its web server is called www.it.murdoch.edu.au
  - Both servers actually exists on the machine with the name dijkstra.it.murdoch.edu.au

The Web Server

- Recall the diagram of a web page access from the last lecture.
- A Web server is basically a HTTP server
  - listens on port 80
  - to serve request made by clients (usually browsers) – eg. for access to HTML pages.
  - The format of the data exchanged is based on MIME (Multipurpose Internet Mail Extensions).

Front and Back-ends

- In general client and server models, we always have:
  - the *front-end* on the client side - so called because it is the interface front to the real processing.
  - the *back-end* on the server side - so called because in most cases it is where the required processing is done in the background.

The Hypertext Transfer Protocol (HTTP)

- The lastest version is HTTP/1.1, as developed by W3C and IETF.
- In an HTTP interaction, we have one request (in ASCII text), and one response.
- As far as the browser/web server level, each request-response transaction is independent of each other
  - the server/client must use other ways of keeping track of state information if they need to (eg. using cookies).

MIME

- HTTP transfers requests and responses in a format based on MIME.
- Since MIME is a specifications for email, most HTTP messages look like email messages.
- HTTP messages consists of
  - A message body - ie. the content of the request/response.
  - Message headers - ie. control information required to transfer the data

HTTP Requests

- An example:

```
GET /index.html HTTP/1.1
Connection: Keep-Alive
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
image/png, */*
Accept-Charset: iso-8859-1,*,utf-8
Accept-Encoding: gzip
Accept-Language: en
Host: arginine.it.murdoch.edu.au:12345
User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows NT;
DigExt)
```


HTTP Request Line

- The *request-line* in a HTTP request (the first line in the request) consist of
  - a *method* to be executed
  - a resource to execute on (very commonly a web page)
  - the version of HTTP used by the web browser

```
GET  /index.html  HTTP/1.1
```


HTTP Methods

- HTTP/1.1 servers and clients must implement the following 3 methods
  - GET: retrieve a resource from server and send to client.
  - HEAD: same request as GET, but only want the headers in the response.
- 5 other optional methods:
  - POST: send data from client to server (eg. when users fills a HTML form on the client) by appending to a resource.
  - PUT: request a server to replace a resource.
  - DELETE: request a server to remove a resource
  - TRACE: request a loop-back, by having the server send back the whole request message. Used for diagnostics.
  - CONNECT: used by proxies only.
  - OPTIONS: request information about the options available on the server.

HTTP Responses

- An example:

```
HTTP/1.1 200 OK
MIME-Version: 1.0
Server: Apache/1.3.12 (Unix)
Content-Type: text/html
Content-Length: 1234

<HTML>
  <HEAD>
    <TITLE>My Web Page</TITLE>
    <BODY>
      <P>This is my web page</P>
    </BODY>
</HTML>
```

HTTP Response Status Line

- HTTP responses are similar in format to a request, but instead of a Request-line, it has a Status-line to show the status of the response.
- The status line will contain the version of HTTP used by the web server and the result status code, consisting of a number and a text string describing the status.

```
HTTP/1.1   200   OK
```


HTTP Status Codes

- Categories of status codes

Category	Code numbers	Description
Informational	100-199	Application specific
Successful	200-299	Request successful
Redirection	300-399	Client needs to do further action
Client Error	400-499	Problems on client-side
Server Error	500-599	Problems on serve-side

HTTP Status Codes (cont'd)

- Come example common response status codes:

Code #	Code String	Description
200	OK	No error, request succeeded
301	Moved Permanently	Requested resource has moved to another URL permanently
400	Bad Request	The server does not understand the request
403	Forbidden	The client is not allowed to access the resource
404	Not Found	The server cannot find the resource

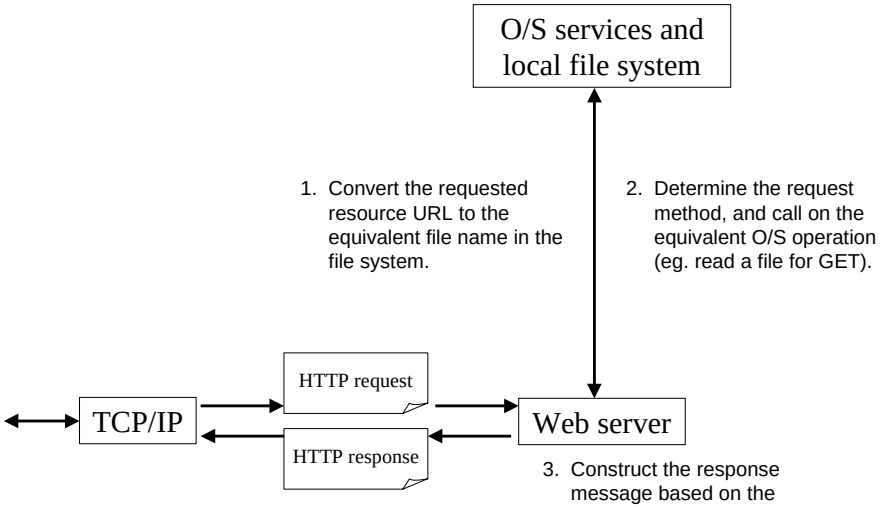
HTTP Header Fields

- Besides the request-line and status-line, HTTP messages also contains other header fields.
- Some are specific to requests, some to responses, and some are not supported by various clients and server.

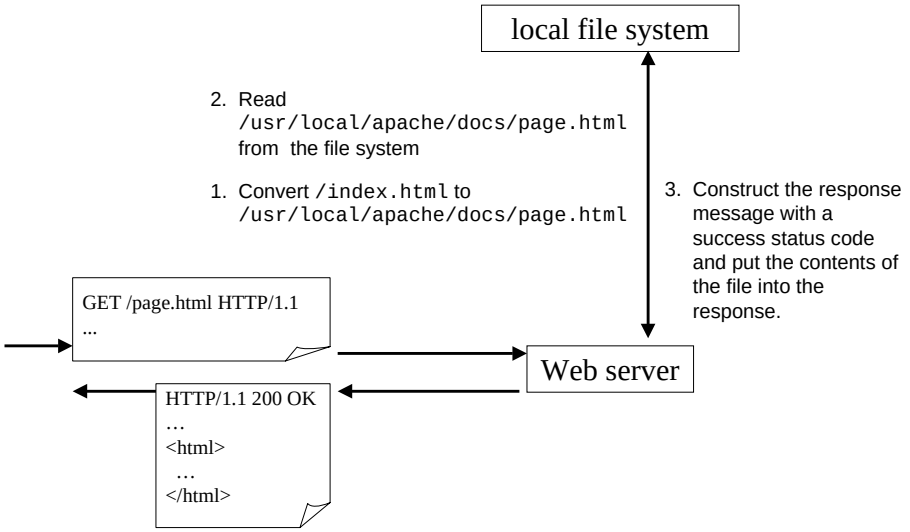
Example Header Fields

- Some example header fields used in most HTTP messages:
  - **User-Agent**: the name and version of the client
  - **Server**: the name and version of the server
  - **Accept**: media types the client is capable of accepting
  - **Allow**: the HTTP methods supported by the server
  - **Content-Type**: the media type of the message body
  - **Expires**: date and time the data in this message becomes invalid
  - **Pragma**: general-purpose field. A commonly used pragma value is "no-cache", which indicates the data is very temporary and should not be cache.

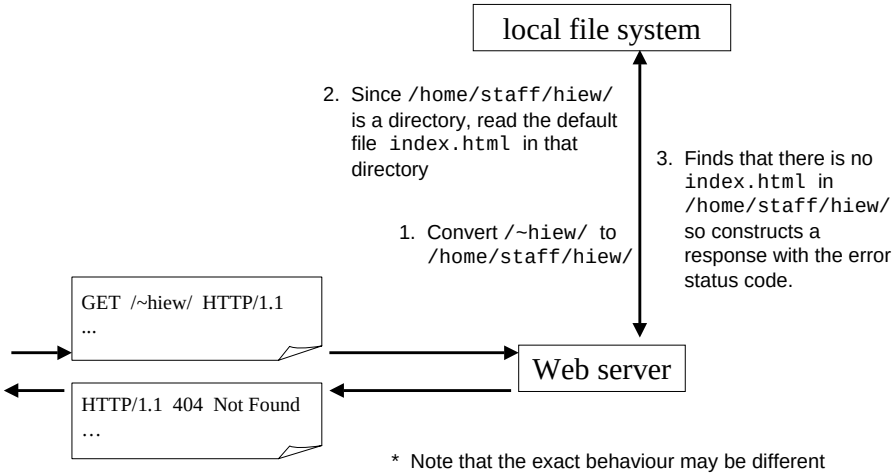
Basic Tasks of a Web Server



Example Task of a Web Server



Example Task of a Web Server



* Note that the exact behaviour may be different depending on how the server is configured.

Down at the TCP level

- If you are interested in what happens at the TCP level, refer to Unit Reader section 2 part 2.2.2.
    - Good to know for understanding of persistent vs non-persistent connections.
    - Requires some knowledge of TCP connection set-up.
    - Not required for this unit.

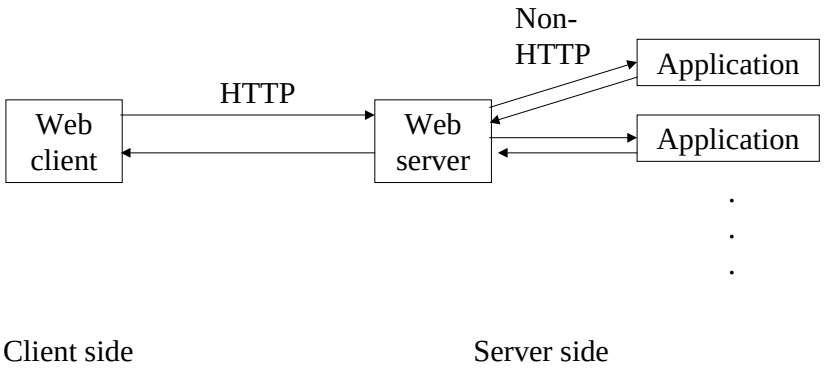
The Common Gateway Interface (CGI)

- A web server has basic operations it can handle, as defined by HTTP, but a client may want to access other non-HTTP services using their HTTP clients (ie. browsers).
- CGI is platform-independent interface specification used to run software in conjunction with an HTTP server
- It acts as a "gateway" between non-HTTP servers/software with the HTTP (web) server.

CGI: The Process

- Process:
    - Client starts a CGI-application, by requesting it through the web server – done by specifying the URL of the application
    - The web server executes the application
    - The web server collects the output of the application and sends in back to the client

CGI: The Process (cont'd)



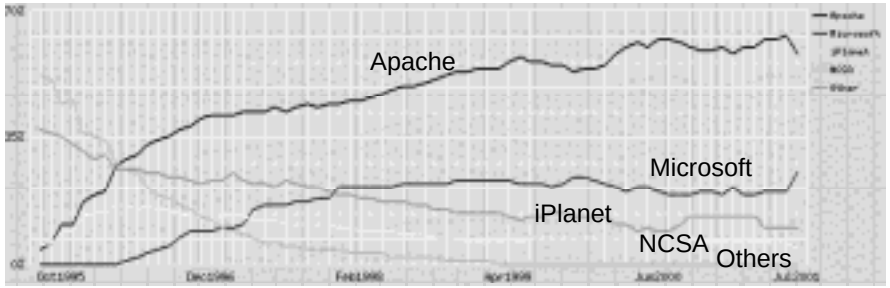
What Does CGI Define?

- HTTP defines the communication between the server and the client requesting the CGI-application – by its normal request/response protocol.
- CGI defines the communication between the HTTP server and the CGI-applications, by giving
  - Command-line parameters
  - Environment variables
  - Input
  - Output

CGI Applications

- Software communication using CGI, but written in scripting/programming languages (such as PERL, Unix shells, C++, Java), are called *CGI scripts/applications*.
- A common example of CGI software are databases.
- Security is NOT defined in CGI. The servers must implement its own security measures when serving CGI requests.

Current Web Server Usage



Source: http://www.netcraft.com/Survey/

Current Web Server Usage

- Approximate current web server usage on the WWW:
  Apache: approx 60%
  Microsoft: approx 27%
  iPlanet: approx 4%
  Zeus: approx 2.5%
- Reference: http://www.netcraft.com/Survey/

References:

- For those interested in further technical details about HTTP
  - http://www.w3.org/Protocols/
  - ftp://ftp.isi.edu/in-notes/rfc2616.txt
- For descriptions and details of various web servers available today:
  - http://serverwatch.internet.com/webservers.html